Fourier Transforms & Spectral Analysis

The purpose of this exercise is to familiarize you with concepts relating to Fourier analysis, sometimes referred to as spectral analysis. Fourier analysis has applications from electrical systems to vibratory systems, data processing to fluid mechanics.

Many functions can be written as an infinite series, such as the Taylor Series. One of the most useful series is the Fourier series. Fourier states that any arbitrary *periodic* function with *finite discontinuities* can be written as the infinite sum of sines and cosines. Equation 1 depicts the basic form of the Fourier Series, where T is the period and the frequency $\omega_0 = \frac{2\pi}{T}$.

$$f(t) = f(t+T) = c_0 + \sum_{n=1}^{\infty}[a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)] \tag{1a}$$

$$= c_0 + \sum_{n=1}^{\infty} A_n e^{in\omega_0 t} \tag{1b}$$

It is often said that a signal has certain "frequency content," meaning that $a_n, b_n \neq 0$ for certain frequencies $\omega_n = n\omega_0$. A signal is said to have frequency content at ω_n.

The Fourier transform is defined as follows:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi\omega t}\,dt \tag{2}$$

Note how similar this definition is to the definition of the Laplace Transform, which you may be more familiar with. It is important to recognize that *no information is gained or lost* when evaluating the Fourier transform. One can transform back and forth between the time and frequency domains without losing any information. Similarly, no information is created. These properties are retained when the Fourier transform is discretized.

The discrete Fourier transform (DFT) is analogous to the continuous formulation:

$$F_k = \sum_{m=1}^{N} f_m \bullet e^{-\frac{i2\pi}{N}(k-1)(m-1)} \tag{3}$$

where N is the total number of discrete points (e.g. data points), f_m is the value of the m^{th} data point, and F_k is k^{th} data point of the transformed function. Time corresponds to *m*; frequency corresponds to *k*. It should be noted, though, that *m* and *k* are simply *labels* for each data point, not the actual value of time and frequency. For example, the first data point (m = 1) may correspond to time zero (t = 0); (k = 1) may correspond to zero frequency (ω = 0).

In practice, Equation 3 is rarely used. It is cumbersome; calculating every term in the series is unnecessary. Instead, the Fast Fourier Transform (FFT) is used. The FFT is simply a more computationally efficient method for calculating Equation 3; it is a type of DFT.

1. **Write your own MATLAB code that will evaluate the Equation 3 for xt1 in dataset1. Compare the time it takes for your code to run compared to the fft() function in MATLAB (fft stands for Fast Fourier Transform). What are some ways you can reduce the amount of time your DFT takes?**

Much like the continuous Fourier transform, the DFT assumes that the function is *periodic*, which may not be the case for a discrete set of points. Furthermore, the DFT violates one of the original assumptions: that the function is finitely discontinuous. By definition, a periodic discrete function is not finitely discontinuous. Violating these assumptions leads to some interesting behavior. For the sake of clarity, the remainder of this document will discuss Fourier analysis in the context of data collection, unless explicitly stated otherwise.

The assumption of periodicity in the time domain requires that the transformed function F_k is also periodic in the frequency domain. As a result, F_k reflects at a certain frequency called the Nyquist frequency. This reflection is the cause of aliasing. Aliasing is when a high frequency signal appears to the measurer to be a lower frequency signal. The Nyquist frequency occurs at half of the sampling frequency, i.e.

$$f_{Nyq} = \frac{1}{2} f_{samp} = \frac{N}{2T}, \tag{4}$$

where the *f* in Equation 4 means frequency, not the function $f(t)$. In other words, any data points beyond the Nyquist frequency are meaningless.

Furthermore, the frequency function reflects to negative frequencies, analogous to how negative time is simply the time before the measurement began. However, we only evaluate the DFT for $m = 1 \rightarrow \infty$. Why is that? In a physical sense, a "negative frequency" is identical to its positive counterpart. In other words, half of the "energy" of the frequency function F_k is lost in the negative frequency domain. It is more physically intuitive to use a frequency domain from $[0, \infty)$ and double the magnitude of the frequency function to account for the lost "energy." It has become convention to double the magnitude when processing DFT results.

2. **Calculate the Nyquist frequency for dataset1. Then, modify the code you wrote for Problem 1 to account for the reflection into the negative frequency domain. Finally, truncate your DFT vector so that only frequencies below the Nyquist frequency are included.**

After Problem 2, your code should look similar to the code in MATLAB's fft() documentation (Figure 1). However, the first line in Figure 1 remains unexplained. The first line is a normalization of the data by the number of data points in the set (L, in Figure 1). Since the DFT is a summation, more data points generally equate to a larger magnitude. Normalization allows us to compare data sets of different sizes. This form of normalization is one convention; it is not the only form of normalization available.

```
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
```

FIGURE 1. Portion of MATLAB fft() documentation.

Furthermore, the first line takes the absolute value of the frequency function (Y, in Figure 1). Before applying the absolute value, the frequency function is complex, containing both magnitude and phase information. The absolute value of the frequency function is a power spectrum of the frequency content; it is real-valued.

Finally, note that the third line in Figure 1 omits the first index value of P1, which should correspond to $\omega_n = 0$. When $\omega_n = 0$, the only content is the mean value of the periodic function (e.g. the DC offset in electrical signals). Since the $\omega_n = 0$ value is not reflected into the negative frequency domain, it does not need to be doubled, thus it is omitted in line 3. At this point, the DFT data should be fully processed for interpretation.

3. **Plot the results from your DFT function and from MATLAB's fft(). Compare. Make sure to scale the frequency axis correctly so it corresponds to correct frequencies (Hint: use the Nyquist frequency as a guideline).**

While the results from Problem 3 are useful, there are a number of other useful tricks for clarifying the results of a DFT/FFT. Two tricks are windowing and zero-padding.

Windows are piecewise functions that are identically zero when outside the data domain but multiply the data by some factor within the data domain. Windows ensure that a set of data meets the a requirement of periodicity: the function begins and ends at the same value. When applying a DFT on a finite set of data, a rectangular window is implicitly being applied. A rectangular window multiplies the function by 1 within the domain; multiplies by zero outside the domain. Other common windows include the Hann/Hamming windows and Blackman windows. A list of various windows can be found at https://en.wikipedia.org/wiki/Window_function.

Windows effectively improve the "periodicity" of a data set, i.e. windows make a data set appear more periodic. The net effect is that peaks can be concentrated (made thinner and taller), making it easier to identify frequencies of interest if peaks are close together. However, the drawback is that frequency resolution is often sacrificed. Frequency resolution is the difference in frequency between two adjacent data points. Rectangular windows have great frequency resolution, but poor peak concentration; Hann windows are a middle ground; Blackman-Harris windows are great at concentrating peaks.

4. **a) Apply an FFT to signal1 in dataset2. Does it appear to have one or two peaks? What window might you use to check? Provide a time-domain signal plot and the FFT plot.**

   b) Apply an FFT to your windowed signal. Are there two peaks? What are their frequencies? Provide a time-domain signal plot and the FFT plot.

Zero padding is a way to make use of the fact that the DFT operates on a finite, discrete signal. The frequency resolution of a signal is directly dependent on the number of samples that were collected. By adding arbitrary zeros to a signal, nothing about the signal is changed except for the number of samples. *No information is created here*, we are simply accessing information that already existed by forcing the DFT to calculate in smaller intervals.

5. **Take the FFT of signal2 in dataset2. Is the resolution high enough to correctly observe the frequency content? Zero pad signal2, then take the FFT again. Are there one or two frequencies in the signal? What are they? Include FFT plots from before and after zero-padding.**

6. **Using the data provided, collected from a vibrating beam, measure the natural frequency with the FFT.**